# Table Documentation

The DataGridComponent provides a comprehensive data table solution with advanced features including sorting, filtering, custom cell templates, and responsive design. It uses a flexible column definition system with content projection for maximum customization.

## How to use

Import the component and its directives, then define your table structure with custom templates.

## Basic Usage

Simple table with basic data display and column definitions.

```
<div class="demo-page">
  <!-- Demo Content -->
  <div class="demo-content">
    <div class="container">
      <!-- Employee Table Section -->
      <div class="demo-section">
        <div class="table-container">
          <aava-data-grid
            [dataSource]="basicData"
            [displayedColumns]="displayedColumns"
            class="styled-data-grid"
          >
            <ng-container avaColumnDef="name">
              <ng-container *avaHeaderCellDef>
                <div class="header-cell">
                  <span class="header-text">Employee Name</span>
                </div>
              </ng-container>
              <ng-container *avaCellDef="let row">
                <div class="data-cell name-cell">
                  <span class="employee-name">{{ row.name }}</span>
                </div>
              </ng-container>
            </ng-container>

            <ng-container avaColumnDef="email">
              <ng-container *avaHeaderCellDef>
                <div class="header-cell">
                  <span class="header-text">Email Address</span>
                </div>
              </ng-container>
              <ng-container *avaCellDef="let row">
                <div class="data-cell email-cell">
                  <span class="email-text">{{ row.email }}</span>
                </div>
              </ng-container>
            </ng-container>

            <ng-container avaColumnDef="department">
              <ng-container *avaHeaderCellDef>
                <div class="header-cell">
                  <span class="header-text">Department</span>
                </div>
              </ng-container>
              <ng-container *avaCellDef="let row">
                <div class="data-cell department-cell">
                  <span class="department-badge">{{ row.department }}</span>
                </div>
              </ng-container>
            </ng-container>

            <ng-container avaColumnDef="status">
              <ng-container *avaHeaderCellDef>
                <div class="header-cell">
                  <span class="header-text">Status</span>
                </div>
              </ng-container>
              <ng-container *avaCellDef="let row">
                <div class="data-cell status-cell">
                  <aava-tag
```

```
                    [label]="row.status"
                    [color]="getStatusColor(row.status)"
                    size="sm"
                  ></aava-tag>
                </div>
              </ng-container>
            </ng-container>
          </aava-data-grid>
        </div>
      </div>
    </div>
  </div>
</div>


---

 basicData = [
    {
      id: 1,
      name: 'Alice Johnson',
      email: 'alice.johnson@example.com',
      department: 'Engineering',
      status: 'Active',
    },
    {
      id: 2,
      name: 'Bob Smith',
      email: 'bob.smith@example.com',
      department: 'Marketing',
      status: 'Active',
    },
    {
      id: 3,
      name: 'Carlos Martinez',
      email: 'carlos.martinez@example.com',
      department: 'Sales',
      status: 'Pending',
    },
    {
      id: 4,
      name: 'Diana Lee',
      email: 'diana.lee@example.com',
      department: 'Engineering',
      status: 'Inactive',
    },
    {
      id: 5,
      name: 'Ethan Brown',
      email: 'ethan.brown@example.com',
      department: 'HR',
      status: 'Active',
    },
  ];

  displayedColumns = ['name', 'email', 'department', 'status'];

  /**
   * Get the appropriate color for status tags
   */
```

```
getStatusColor(
  status: string
): 'success' | 'warning' | 'error' | 'info' | 'default' {
  switch (status.toLowerCase()) {
    case 'active':
      return 'success';
    case 'pending':
      return 'warning';
    case 'inactive':
      return 'error';
    default:
      return 'default';
  }
}
```

## Sorting

Table with sortable columns and visual sort indicators.

```
<div class="demo-content">
  <div class="demo-section">
    <div class="demo-card">
      <div class="card-content">
        <aava-data-grid
          [dataSource]="employeeData"
          [displayedColumns]="displayedColumns"
        >
          <ng-container avaColumnDef="name" [sortable]="true">
            <ng-container *avaHeaderCellDef>Employee Name</ng-container>
            <ng-container *avaCellDef="let row">{{ row.name }}</ng-container>
          </ng-container>

          <ng-container avaColumnDef="position" [sortable]="true">
            <ng-container *avaHeaderCellDef>Position</ng-container>
            <ng-container *avaCellDef="let row"
              >{{ row.position }}</ng-container
            >
          </ng-container>

          <ng-container avaColumnDef="salary" [sortable]="true">
            <ng-container *avaHeaderCellDef>Annual Salary</ng-container>
            <ng-container *avaCellDef="let row"
              >${{ row.salary | number }}</ng-container
            >
          </ng-container>

          <ng-container avaColumnDef="experience" [sortable]="true">
            <ng-container *avaHeaderCellDef>Experience (Years)</ng-container>
            <ng-container *avaCellDef="let row"
              >{{ row.experience }} years</ng-container
            >
          </ng-container>

          <ng-container avaColumnDef="joinDate" [sortable]="true">
            <ng-container *avaHeaderCellDef>Join Date</ng-container>
            <ng-container *avaCellDef="let row"
              >{{ row.joinDate | date }}</ng-container
            >
          </ng-container>

          <ng-container avaColumnDef="department">
            <ng-container *avaHeaderCellDef>Department</ng-container>
            <ng-container *avaCellDef="let row"
              >{{ row.department }}</ng-container
            >
          </ng-container>
        </aava-data-grid>
      </div>
    </div>
  </div>
</div>


---

employeeData = [
  {
    id: 1,
    name: "Alice Johnson",
```

```
  position: "Senior Developer",
  salary: 95000,
  joinDate: "2020-03-15",
  experience: 8,
  department: "Engineering",
},
{
  id: 2,
  name: "Bob Smith",
  position: "Marketing Manager",
  salary: 75000,
  joinDate: "2019-07-22",
  experience: 6,
  department: "Marketing",
},
{
  id: 3,
  name: "Carlos Martinez",
  position: "Sales Representative",
  salary: 55000,
  joinDate: "2021-11-08",
  experience: 3,
  department: "Sales",
},
{
  id: 4,
  name: "Diana Lee",
  position: "UX Designer",
  salary: 70000,
  joinDate: "2020-09-12",
  experience: 5,
  department: "Design",
},
{
  id: 5,
  name: "Ethan Brown",
  position: "Data Analyst",
  salary: 65000,
  joinDate: "2022-01-30",
  experience: 2,
  department: "Analytics",
},
{
  id: 6,
  name: "Fiona Green",
  position: "Project Manager",
  salary: 85000,
  joinDate: "2018-05-10",
  experience: 9,
  department: "Operations",
},
{
  id: 7,
  name: "George Wang",
  position: "DevOps Engineer",
  salary: 90000,
  joinDate: "2019-12-03",
  experience: 7,
  department: "Engineering",
},
```

```
  {
    id: 8,
    name: "Hannah Kim",
    position: "Content Writer",
    salary: 45000,
    joinDate: "2021-08-15",
    experience: 1,
    department: "Marketing",
  },
];

displayedColumns = ["name", "position", "salary", "experience", "joinDate"];

salesData = [
  { month: "January", revenue: 125000, orders: 340, conversion: 3.2 },
  { month: "February", revenue: 135000, orders: 385, conversion: 3.8 },
  { month: "March", revenue: 142000, orders: 420, conversion: 4.1 },
  { month: "April", revenue: 128000, orders: 365, conversion: 3.5 },
  { month: "May", revenue: 155000, orders: 445, conversion: 4.3 },
  { month: "June", revenue: 168000, orders: 478, conversion: 4.6 },
];

salesColumns = ["month", "revenue", "orders", "conversion"];
```

## Filtering

Advanced filtering capabilities with multiple filter conditions and operators.

```
<div class="demo-content">
  <div class="demo-section">
    <div class="demo-card">
      <div class="card-content">
        <aava-data-grid
          [dataSource]="inventoryData"
          [displayedColumns]="inventoryColumns"
        >
          <ng-container avaColumnDef="sku" [filter]="true" [sortable]="true">
            <ng-container *avaHeaderCellDef>SKU</ng-container>
            <ng-container *avaCellDef="let row">
              <code class="sku-code">{{ row.sku }}</code>
            </ng-container>
          </ng-container>

          <ng-container
            avaColumnDef="product"
            [filter]="true"
            [sortable]="true"
          >
            <ng-container *avaHeaderCellDef>Product Name</ng-container>
            <ng-container *avaCellDef="let row">{{ row.product }}</ng-container>
          </ng-container>

          <ng-container
            avaColumnDef="category"
            [filter]="true"
            [sortable]="true"
          >
            <ng-container *avaHeaderCellDef>Category</ng-container>
            <ng-container *avaCellDef="let row">
              <span class="category-tag">{{ row.category }}</span>
            </ng-container>
          </ng-container>

          <ng-container avaColumnDef="stock" [filter]="true" [sortable]="true">
            <ng-container *avaHeaderCellDef>Stock</ng-container>
            <ng-container *avaCellDef="let row">
              <span
                class="stock-indicator"
                [class]="row.stock < 50 ? 'low-stock' : 'normal-stock'"
              >
                {{ row.stock }} units
              </span>
            </ng-container>
          </ng-container>

          <ng-container avaColumnDef="price" [filter]="true" [sortable]="true">
            <ng-container *avaHeaderCellDef>Price</ng-container>
            <ng-container *avaCellDef="let row">${{ row.price }}</ng-container>
          </ng-container>
        </aava-data-grid>
      </div>
    </div>
  </div>
</div>


---
customerData = [
```

```
{
  id: 1,
  name: "Alice Johnson",
  email: "alice.johnson@techcorp.com",
  company: "TechCorp Inc.",
  status: "Active",
  location: "New York",
  industry: "Technology",
  revenue: 250000,
},
{
  id: 2,
  name: "Bob Smith",
  email: "bob.smith@retail.co",
  company: "Retail Solutions Co.",
  status: "Inactive",
  location: "Los Angeles",
  industry: "Retail",
  revenue: 180000,
},
{
  id: 3,
  name: "Carlos Martinez",
  email: "carlos@manufacturing.biz",
  company: "Manufacturing Plus",
  status: "Pending",
  location: "Chicago",
  industry: "Manufacturing",
  revenue: 320000,
},
{
  id: 4,
  name: "Diana Lee",
  email: "diana.lee@healthsys.org",
  company: "HealthSys Group",
  status: "Active",
  location: "Houston",
  industry: "Healthcare",
  revenue: 420000,
},
{
  id: 5,
  name: "Ethan Brown",
  email: "ethan@finance.net",
  company: "Finance Solutions",
  status: "Active",
  location: "Miami",
  industry: "Finance",
  revenue: 380000,
},
{
  id: 6,
  name: "Fiona Green",
  email: "fiona.green@education.edu",
  company: "Education First",
  status: "Inactive",
  location: "Seattle",
  industry: "Education",
  revenue: 95000,
},
```

```
  {
    id: 7,
    name: "George Wang",
    email: "george@consulting.pro",
    company: "Consulting Experts",
    status: "Active",
    location: "Boston",
    industry: "Consulting",
    revenue: 275000,
  },
  {
    id: 8,
    name: "Hannah Kim",
    email: "hannah.kim@media.tv",
    company: "Media Productions",
    status: "Pending",
    location: "San Francisco",
    industry: "Media",
    revenue: 150000,
  },
  {
    id: 9,
    name: "Ian Davis",
    email: "ian@logistics.freight",
    company: "Logistics Express",
    status: "Active",
    location: "Denver",
    industry: "Logistics",
    revenue: 200000,
  },
  {
    id: 10,
    name: "Julia Roberts",
    email: "julia@realestate.homes",
    company: "Real Estate Pros",
    status: "Active",
    location: "Phoenix",
    industry: "Real Estate",
    revenue: 310000,
  },
];

displayedColumns = [
  "name",
  "email",
  "company",
  "status",
  "location",
  "industry",
];

inventoryData = [
  {
    sku: "TECH-001",
    product: "Wireless Mouse",
    category: "Electronics",
    stock: 150,
    price: 29.99,
  },
  {
```

```
      sku: "TECH-002",
      product: "Bluetooth Keyboard",
      category: "Electronics",
      stock: 85,
      price: 79.99,
    },
    {
      sku: "BOOK-001",
      product: "JavaScript Handbook",
      category: "Books",
      stock: 45,
      price: 34.95,
    },
    {
      sku: "FURN-001",
      product: "Ergonomic Chair",
      category: "Furniture",
      stock: 12,
      price: 299.99,
    },
    {
      sku: "TECH-003",
      product: "USB-C Cable",
      category: "Electronics",
      stock: 200,
      price: 14.99,
    },
    {
      sku: "BOOK-002",
      product: "Design Principles",
      category: "Books",
      stock: 28,
      price: 42.5,
    },
    {
      sku: "FURN-002",
      product: "Standing Desk",
      category: "Furniture",
      stock: 8,
      price: 449.99,
    },
    {
      sku: "TECH-004",
      product: "Laptop Stand",
      category: "Electronics",
      stock: 67,
      price: 89.99,
    },
  ];

  inventoryColumns = ["sku", "product", "category", "stock", "price"];
```

## Features

## Flexible Column System

- Content projection-based column definitions
- Custom header and cell templates

• Configurable sorting and filtering per column
• Dynamic column visibility

## Advanced Sorting

• Multi-column sorting support
• Visual sort indicators (ascending/descending)
• Configurable sort behavior per column
• Sort state management

## Powerful Filtering

• Multiple filter conditions and operators
• Real-time filtering with search
• Filter panel with advanced options
• Clear and apply filter actions

## Custom Templates

• Flexible cell content templates
• Custom header templates
• Template context with row data and index
• Support for complex cell content

## Responsive Design

• Horizontal scrolling for wide tables
• Mobile-friendly design
• Adaptive column sizing
• Touch-optimized interactions

## Performance Optimized

• OnPush change detection strategy
• Efficient data handling
• Optimized rendering
• Memory management

## API Reference

## Inputs

| Property | Type | Default | Description |
|---|---|---|---|
| dataSource | any[] | [] | Array of data objects to display in the table |
| displayedColumns | string[] | [] | Array of column names to display |

## Outputs

| Property | Type | Description |
|---|---|---|
| dataSorted | EventEmitter | Emitted when data is sorted with sorted data |

## Directives

## AvaColumnDefDirective

| Property | Type | Default | Description |
|---|---|---|---|
| avaColumnDef | string | - | Column name/identifier (required) |
| sortable | boolean | false | Enable sorting for this column |
| filter | boolean | false | Enable filtering for this column |

## AvaHeaderCellDefDirective

| Property | Type | Description |
|---|---|---|
| Template | TemplateRef | Template for custom header cell content |

## AvaCellDefDirective

| Property | Type | Description |
|---|---|---|
| Template | TemplateRef | Template for custom cell content with context |

## Interfaces

## Methods

| Method | Parameters | Description |
|---|---|---|
| onSort() | column: AvaColumnDefDirective | Handle column sorting |
| applySort() | None | Apply current sort to data |

| Method | Parameters | Description |
|---|---|---|
| applyFilter() | columnName: string, event: Event | Apply filter to specific column |
| clearFilter() | columnName: string, event: any | Clear filter for specific column |
| openPanel() | columnName: string, event: any | Open filter panel for column |
| checkForOpen() | columnName: string | Check if filter panel is open for column |

## Properties

| Property | Type | Description |
|---|---|---|
| sortColumn | string \| null | Currently sorted column |
| sortDirection | 'asc' \| 'desc' \| '' | Current sort direction |
| sortedData | any[] | Currently sorted and filtered data |
| filterColumn | Array<{column: string, type: string, value: any, open: boolean}> | Active filters |
| defaultFilterConditions | FilterCondition[] | Available filter conditions |

## CSS Custom Properties

The component uses CSS custom properties for dynamic styling:

## Container Properties

| Property | Description |
|---|---|
| --grid-font-family-body | Font family for table content |
| --grid-text-color | Text color for table content |
| --grid-background-color-odd | Background color for odd rows |
| --grid-background-color-even | Background color for even rows |
| --grid-border | Border color for grid elements |

## Table Properties

| Property | Description |
|---|---|
| --table-border | Border color for table elements |

## CSS Classes

The component uses CSS classes for styling and state management:

## Container Classes

| Class | Description |
|---|---|
| .ava-data-table-wrapper | Main table container |
| .data-table-wrapper | Inner table wrapper with scrolling |
| .ava-data-table | Main table element |

## Cell Classes

| Class | Description |
|---|---|
| .cell-wrapper | Header cell content wrapper |
| .grid-column-container | Column header container |
| .filter | Filter icon container |
| .filter-wrapper | Filter panel container |
| .default-filter-actions | Filter action buttons container |
| .cell-link | Link styling within cells |

## State Classes

| Class | Description |
|---|---|
| .sort-icon | Sort indicator icon |
| Various pseudo-classes | Hover and focus states |

## Best Practices

## Data Structure

- Use consistent data structure across all rows
- Ensure column names match displayedColumns array
- Provide meaningful default values for missing data
- Optimize data for sorting and filtering

## Column Definitions

- Use descriptive column names
- Enable sorting only for relevant columns

• Enable filtering for searchable data
• Provide meaningful header labels

## Custom Templates

• Keep cell templates simple and focused
• Use template context for row data access
• Implement proper error handling in templates
• Consider accessibility in custom content

## Performance

• Limit data size for optimal performance
• Use OnPush change detection strategy
• Implement virtual scrolling for large datasets
• Optimize filter and sort operations

## Accessibility

• Provide proper ARIA labels
• Ensure keyboard navigation support
• Use semantic HTML structure
• Maintain color contrast ratios

## Responsive Design

• Test table on various screen sizes
• Implement horizontal scrolling for wide tables
• Consider mobile-specific interactions
• Optimize touch targets for mobile

## Accessibility

## ARIA Support

• Proper table semantics
• Sort and filter announcements
• Screen reader friendly navigation
• Status updates for dynamic content

## Keyboard Navigation

• Tab navigation through table elements
• Arrow key navigation between cells
• Enter/Space activation for actions
• Escape key for closing panels

## Focus Management

• Clear focus indicators
• Logical tab order
• Focus restoration after actions
• Focus trapping in modals/panels

## Screen Reader Support

• Descriptive labels for actions
• Context information for data
• Status announcements
• Clear navigation structure

## Browser Support

• Modern Browsers : Full support for all features
• CSS Grid/Flexbox : Required for layout
• ES6+ Features : Required for component functionality
• Template Ref : Required for content projection
• Change Detection : OnPush strategy support